Remote user authentication using public information*

Chris J. Mitchell

Mobile VCE Research Group, Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
C.Mitchell@rhul.ac.uk

Abstract. A method for remote user authentication is proposed that requires only public information to be stored at the verifying host. Like the S/KEY scheme, the new technique uses only symmetric cryptography and is resistant to eavesdropping, but, unlike S/KEY, it is resistant to host impersonation attacks. The avoidance of asymmetric cryptographic techniques makes the scheme appropriate for low cost user authentication devices.

1 Introduction

Authentication of remote users is a problem commonly encountered in distributed computing environments. It is a problem addressed by the S/KEY user authentication system, details of which have been published as an Internet RFC, [2]. A complete software implementation of S/KEY has also been made publicly available (see [2]).

The S/KEY scheme, which is closely based on a scheme devised by Lamport, [6], has been designed to provide users with 'one-time passwords', which can be used to control user access to remote hosts. Of course, as with any such system, after the user authentication process is complete, i.e. after the one-time password has been sent across the network, no protection is offered against subversion of the link by third parties. This fact is pointed out in [2]. Indeed it is stated there that the S/KEY scheme 'does not protect a network eavesdropper from gaining access to private information, and does not provide protection against "inside" jobs or against active attacks where the potential intruder is able to intercept and modify the packet stream'.

It is further claimed that S/KEY 'is not vulnerable to eavesdropping/replay attacks'. Unfortunately, as has been pointed out by a number of authors, (see, for

* The work reported in this paper has formed part of the Software Based Systems area of the Core 2 Research Programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE, www.mobilevce.com, whose funding support, including that of the EPSRC, is gratefully acknowledged. More detailed technical reports on this research are available to Industrial Members of Mobile VCE.

example, [8] or Note 10.7 of [7]) depending on the definition of 'replay attack', the S/KEY scheme can fail to provide this property. Specifically, the S/KEY scheme is subject to a 'host impersonation' attack, where the false host can obtain information from the remote user which can be used to impersonate the remote user to the genuine host at some later occasion.

One major advantage of S/KEY over other user authentication schemes is that it only requires the verifying host to store public information about the remote user. Knowledge of this public information is not in itself sufficient to enable a third party to masquerade as the remote user, although if the remote user's secret key is poorly chosen then the public information would enable a brute force search to be performed. The purpose of this paper is to propose an alternative user authentication scheme which retains this major advantage of S/KEY, but which resists host impersonation attacks. It also has the practical implementation advantage that it uses only symmetric cryptography, an important issue if the remote user's secret key is stored in a device (e.g. a cheap smart card) with limited computational power.

2 The new scheme

We suppose that the host H and remote user U have an initial secure session during which the U supplies H with trusted public information. We therefore divide our description into two phases, 'set up' and 'use'.

Prior to these phases, two system parameters t and r are selected, where t and r are positive integers satisfying $r < t$. The choice of these values affects the security of the scheme (see Section 3 below). A method for computing MACs (Message Authentication Codes) must also be agreed; this could be HMAC or a block cipher based CBC-MAC — see, for example, ISO/IEC 9797, [4, 5]. Whatever method is chosen must be resistant to both key recovery and forgery attacks. In fact, we require resistance to a slightly generalised version of key recovery. Key recovery attacks normally involve an attacker using a number of (message,MAC) pairs to find the key used to generate the MACs. Here, we do not wish the attacker to be able to find *any* key which maps a given message to a given MAC, regardless of whether or not it was the actual key used. By choosing the algorithm and algorithm parameters carefully, it should be possible to achieve the desired attack resistance.

For the purposes of the discussion below we write $M_K(X)$ for the MAC computed on data X using secret key K.

2.1 Set up phase

The remote user U chooses a set of t secret keys for the MAC algorithm, denotes by $K_1, K_2, \ldots, K_t$. U then chooses a random data string X and computes

$$V_i = M_{K_i}(X)$$

for every i $(1 \leq i \leq t)$. U then:

– passes the values $V_1, V_2, \ldots, V_t$ and X to H, and
– securely stores $K_1, K_2, \ldots, K_t$ and X.

H securely stores $V_1, V_2, \ldots, V_t$ and X as the public verification information for U. The integrity of this information must be preserved, but secrecy is not required.

2.2 Use of the scheme

We now suppose that U wishes to authenticate him/herself to host H. The process operates in a series of steps, as follows. Note that if, at any point, one of the checks made by the receiver of a message fails, then, unless otherwise stated, the entire protocol run is deemed to have failed. This assumption applies throughout this paper.

1. H first sends X to U.
2. U first checks the correctness of X, in case of loss of synchronisation between U and H. If X is incorrect then, in certain circumstances, U may check the previous value of X to see if synchronisation can be restored (this possibility is discussed further in Section 3). U then chooses a new set of t secret keys: $K'_1, K'_2, \ldots, K'_t$ and selects a new random value X'. U also computes two sequences of t values:

$$V'_i = M_{K'_i}(X'),\ W'_i = M_{K_i}(V'_1||V'_2||\cdots||V'_t),\quad (1 \le i \le t)$$

   where here, as throughout, $||$ denotes concatenation of data items.
   U now sends $(W'_1, W'_2, \ldots, W'_t)$ to H.
3. H then chooses a random r-subset of $\{1, 2, \ldots, t\}$, say $\{i_1, i_2, \ldots, i_r\}$, and sends this subset to U.
4. U now sends the r secret keys $K_{i_1}, K_{i_2}, \ldots, K_{i_r}$ to H, as well as the set of t values $V'_1, V'_2, \ldots, V'_t$ and the value X'. U now replaces the stored values $X, K_1, K_2, \ldots, K_t$ with $X', K'_1, K'_2, \ldots, K'_t$. (In certain cases, U may retain the 'old' values, as discussed below).
5. H now verifies the r MAC values $V_{i_1}, V_{i_2}, \ldots, V_{i_r}$ using the set of r keys supplied by U and the stored value of X. If *all* these values are correct, then H also verifies the r MAC values $W'_{i_1}, W'_{i_2}, \ldots, W'_{i_r}$ using the set of r keys supplied by U and the values $V'_1, V'_2, \ldots, V'_t$ supplied by U previously. If all these MACs are also correct, then H accepts U as valid, and replaces $X, V_1, V_2, \ldots, V_t$ with $X', V'_1, V'_2, \ldots, V'_t$.

3 Discussion and analysis

We now consider practical and security aspects of the scheme. Observe that elements of the scheme are very similar to the one time signature (OTS) scheme of Rabin [9] (see also section 11.6 of [7]). Indeed, one way of looking at the scheme above is to regard the set up phase as key generation for an OTS scheme, as a

result of which the host is equipped with the public key. One iteration of the protocol consists of the user generating a new OTS key pair, signing the new public key with the existing OTS private key, and the host then performing a verification process. On completion the host has a copy of the new OTS public key, ready to start the process again.

The scheme also has features in common with the 'Guy Fawkes' protocol of Anderson et al. [1]. However, the scheme nevertheless has properties distinct from previously specified protocols.

3.1 Choices for t and r

We first consider how t and r should be chosen. To avoid certain attacks, we wish to choose these values so that the probability of a third party successfully guessing the subset $\{i_1, i_2, \ldots, i_r\}$ in advance is negligible. That is we wish to arrange things so that $1/\binom{t}{r}$ is negligible.

Given that we wish to minimise t (to minimise the storage and bandwidth requirements) then this probability is minimised by choosing $r = \lfloor t/2 \rfloor$, since $\binom{t}{\lfloor t/2 \rfloor} \geq \binom{t}{i}$ for all i. Also, since $\sum_{i=0}^{t} \binom{t}{i} = 2^t$, we immediately have that $\binom{t}{\lfloor t/2 \rfloor} > 2^t/(t+1)$ if $t > 1$.

Hence, if we wish to guarantee that the probability of successfully guessing the subset is at most 10^{-9} say, then choosing $t \geq 35$ will suffice.

3.2 Host impersonation attacks

Suppose a third party, E say, wishes to impersonate H to U with a view to learning enough to impersonate U to H at some subsequent time. In step 3 of the protocol, E can only choose a random r-subset of $\{1, 2, \ldots, t\}$, and E will then learn a set of r of the secret keys. However, at a later stage, when E impersonates U to H, E will only be successful if he/she knows all the keys in the subset chosen by H. The odds of this will be acceptably small as long as t and r are chosen appropriately (see the discussion in Section 3.1).

3.3 Man in the middle attacks

Of course, as with any authentication protocol, it will always be possible for a third party E to simply sit between U and H in the communications channel, and relay messages between the two. This only becomes a genuine threat if E is able to change some part of the messages, and/or to re-order them in some way. We now look at the various messages in turn, to see if this is possible.

- In step 2, E could change some or all of the MAC values W_i'. However, given that at this stage E will not know any of the keys K_i, the probability that the modified values will be correct is negligibly small (since we assume that forgery attacks are not feasible).
- In step 3, E could change the subset, but then the set of keys returned in step 4 will not be correct.

– In step 4, E could modify some or all of the secret keys K_{i_j} and/or some or all of the MAC values V_i'. Successfully changing the values V_i' would require knowledge of the keys K_i', but none of these have yet been divulged by U. Changing the secret keys K_{i_j} is prevented by the fact that H can check them using the values V_{i_j}. (Changing these verification MACs would have required knowledge of the previous set of keys, and changing these previous keys would have required changing the previous verification MACs, and so on).

3.4 Denial of service attacks

There is, of course, a simple and effective 'denial of service' attack against the described protocol. A third party E can simply engage in the protocol with U by impersonating H. When U tries to authenticate him/herself to the genuine H, U will have a different value of X to that sent by H in the first step of the protocol.

There are two main ways in which this can be dealt with. Firstly, U could simply abandon the attempt to authenticate to H, and arrange for the system to be re-initialised. Secondly, U could retain 'old' values of X (along with the associated set of keys) and use them to complete the authentication protocol. However, such a process has very serious dangers, depending on the choices of t and r.

With r set to $\lfloor t/2 \rfloor$, even doing the process twice would completely destroy the system security. A malicious third party E could impersonate H to U twice, using two disjoint r-subsets of $\{1, 2, \ldots, t\}$. This would mean that E would obtain all of the keys $K_1, K_2, \ldots, K_t$ (or all but one of them if t is odd). As a result, E would be able to impersonate U to H any number of times.

Hence if we are to allow the same key set to be used more than once then r and t need to be chosen appropriately. Also, U needs to implement a counter to limit the number of times any particular key set is used for the authentication process. The limit for this counter will be determined by the choices for t and r. This issue is discussed further in the next section.

3.5 Resynchronisation

As we have just seen, one way of limiting the impact of denial of service attacks by malicious third parties impersonating the host, is to allow a key set to be used more than once. This may also be necessary if the authentication process between user and host fails part way through, e.g. because of a communications failure.

If a key set is to be used up to a maximum of c times (this being enforced by the counter held by U) then it should be the case then any party with knowledge of c different random r-subsets of the set of t keys $K_1, K_2, \ldots, K_t$ should have a negligible probability of knowing all the members of another randomly chosen r-subset of keys.

To compute the necessary probabilities we make some simplifying assumptions (pessimistic from the point of view of the legitimate system users). We suppose that, by bad luck or by host impersonation, all the c different r-subsets are mutually disjoint. Thus we require that the probability that a randomly chosen r-subset of $\{1, 2, \ldots, t\}$ does not contain any element from a specified subset of size $t - cr$ shall be small. The following result, the proof of which is elementary, is therefore relevant.

Lemma 1. *Suppose c, r and t are positive integers satisfying $r(c+1) < t$. If S is a subset of $\{1, 2, \ldots, t\}$ of size cr, then the probability that R, a randomly chosen r-subset of $\{1, 2, \ldots, t\}$, is a subset of S is equal to*

$$\frac{\binom{cr}{r}}{\binom{t}{r}}.$$

We thus require that c, r and t should be chosen so that

$$\frac{cr(cr-1)\ldots(cr-r+1)}{t(t-1)\cdots(t-r+1)},$$

which is bounded above by $(cr/t)^r$, is small. As an example we can put $r = 32$ and $t = 64c$, and we are then guaranteed that the probability of a successful attack is less than 2^{-32}.

4 A streamlined version of the protocol

In the protocol presented above, steps 1 and 2 can be merged with steps 3 and 4 respectively, to give a two-pass protocol. This is at the cost of increasing long-term storage requirements. The modified protocol operates as follows.

4.1 Set up

The remote user U chooses two sets of t secret keys for the MAC algorithm, the *current set*, denoted by $K_1, K_2, \ldots, K_t$, and the *pending set*, denoted by $K'_1, K'_2, \ldots, K'_t$. U chooses two random data strings used as *key set indicators*, denoted by X and X' (for the current and pending key sets).

U now computes verification MACs for both the current and pending key sets as

$$V_i = M_{K_i}(X) \quad \text{and } V'_i = M_{K'_i}(X')$$

for every i $(1 \le i \le t)$. U also computes a further set of t MACs

$$W'_i = M_{K_i}(V'_1||V'_2||\cdots||V'_t), \quad (1 \le i \le t).$$

U then:

- passes the two sets of verification values and the corresponding key set indicators $(V_1, V_2, \ldots, V_t, X)$ and $(V'_1, V'_2, \ldots, V'_t, X')$ to H,

– passes the t MACs $(W'_1, W'_2, \ldots, W'_t)$ to H, and
– securely stores the two key sets with their respective indicators, i.e. stores $(K_1, K_2, \ldots, K_t, X)$ and $(K'_1, K'_2, \ldots, K'_t, X')$.

H securely stores the information received from U. The integrity of this information must be preserved, but secrecy is not required.

4.2 Use of the scheme

Suppose that U wishes to authenticate him/herself to host H. The process operates as follows.

1. H chooses a random r-subset of $\{1, 2, \ldots, t\}$, say $\{i_1, i_2, \ldots, i_r\}$, and sends this subset to U along with the current key set indicator X.
2. U first checks the correctness of X, in case of loss of synchronisation between U and H. If X is incorrect then, in certain circumstances, U may check the previous value of X to see if synchronisation can be restored (as discussed in Section 3).
   U then chooses a new set of t secret keys: $K''_1, K''_2, \ldots, K''_t$ and selects a new random key set indicator X''. U also computes two sequences of t values:

$$V''_i = M_{K''_i}(X'), \;\; W''_i = M_{K'_i}(V''_1 || V''_2 || \cdots || V''_t), \;\; (1 \leq i \leq t).$$

   U now sends X'', $(V''_1, V''_2, \ldots, V''_t)$ and $(W''_1, W''_2, \ldots, W''_t)$ to H. U also sends the r secret keys $K_{i_1}, K_{i_2}, \ldots, K_{i_r}$ to H. U now replaces the stored values of
   – $X, K_1, K_2, \ldots, K_t$ with $X', K'_1, K'_2, \ldots, K'_t$, and
   – $X', K'_1, K'_2, \ldots, K'_t$ with $X'', K''_1, K''_2, \ldots, K''_t$.
3. H now verifies the r MAC values $V_{i_1}, V_{i_2}, \ldots, V_{i_r}$ using the set of r keys supplied by U and the stored value of X. If *all* these values are correct, then H also verifies the r MAC values $W'_{i_1}, W'_{i_2}, \ldots, W'_{i_r}$ using the set of r keys supplied by U and the stored values $V'_1, V'_2, \ldots, V'_t$. If all these MACs are also correct, then H accepts U as valid, and replaces:
   – $X, V_1, V_2, \ldots, V_t$ with $X', V'_1, V'_2, \ldots, V'_t$,
   – $X', V'_1, V'_2, \ldots, V'_t$ with $X'', V''_1, V''_2, \ldots, V''_t$, and
   – $W'_1, W'_2, \ldots, W'_t$ with $W''_1, W''_2, \ldots, W''_t$.

5 Implementation issues

We now consider certain practical implementation issues for the protocol.

5.1 Complexity

We start by considering the storage, computation and communications complexity of the scheme as described in Section 2.

– *Storage:* the requirements for the host are to store t MACs and a random value; the requirements for the user are to store t keys. During execution of the protocol, the remote user and host must both store a further $2t$ MACs, and the user must also temporarily store an additional t keys. Note that, for the streamlined scheme of Section 4, the long term storage requirements for host and user increase to $3t$ MACs and $2t$ secret keys respectively. Note also that, if the user retains 'old' key sets for resynchronisation purposes, then this will be at a cost of t keys, a random value and a usage counter for each retained old set.
– *Computation:* the host verifies $2r$ MACs and chooses one random r-subset of $\{1, 2, \ldots, t\}$, and the user generates $2t$ MACs.
– *Communications:* the user sends the host a total of $2t$ MACs, one random value and r secret keys, and the host sends the user one r-subset of $\{1, 2, \ldots, t\}$.

To see what this might mean in practice, suppose we wish to use the basic scheme (of Section 2) in such a way that a particular key set can be used up to $c = 3$ times, and that the user retains one 'old' key set for resynchronisation purposes. We thus choose $r = 32$ and $t = 196$. Suppose that the MAC in use is HMAC based on SHA-1 (see [5] and [3]) with MACs and keys of 160 bits each; suppose also that X contains 128 bits. Then the user must store $2t$ keys, two random values and a counter (which we ignore since it will take negligible space) — this amounts to $392 \times 20 + 32$ bytes, i.e. just under 8 kbytes, with an additional 12 kbytes of short term storage needed during protocol execution). The host must store approximately 4 kbytes of MACs, with an additional 8 kbytes of short term storage needed during protocol execution. During use of the protocol the user will need to compute 392 MACs and the host will need to compute 64 MACs. The total data to be exchanged between host and user during the protocol amounts to around 8.5 kbytes.

Note that the values X do not need to be random or unpredictable - U could generate the values using a modest-sized counter (e.g. of 4 bytes). This would save a small amount of communications and storage costs. It is also tempting to try and reduce the MAC lengths. However, significant length reductions are not possible since, as noted in Section 2, we do not wish the attacker to be able to find *any* key which maps a given message to a given MAC, regardless of whether or not it was the actual key used. This implies that MACs need to be of length close to that assumed immediately above, although a reasonable saving achievable by reducing MACs to, say, 10 bytes is not infeasible. Whilst such modifications will not significantly reduce user storage requirements, the communications requirements are almost halved, as are the host storage requirements.

5.2 A security improvement

In cases where $c > 1$, i.e. where key sets may be used more than once, a malicious entity impersonating the host is free to choose the subsets $\{1, 2, \ldots, r\}$

disjointly so as to learn the maximum number of secret keys. This maximises the (small) probability this malicious entity will have of impersonating the user to the genuine host (see Section 3.5). To avoid this, i.e. to increase the difficulty of launching a host impersonation attack, we can modify the protocol so that neither the remote user nor the host chooses the r-subset $\{i_1, i_2, \ldots, i_r\}$ of $\{1, 2, \ldots, t\}$. This can be achieved by prefixing the protocol of Section 4 with two additional messages, and also making use of an appropriate one-way, collision-resistant hash-function (see, for example, [3]). Note that these two additional messages can be merged with the first two messages of the basic protocol of Section 2.

The revised protocol from Section 4.2 starts as follows:

1. H chooses a random value r_H, of length comparable to the key length in use, computes $h(r_H)$ (where h is a pre-agreed hash-function), and sends $h(r_H)$ to U.
2. U chooses a random value r_U, of length the same as r_H, and sends it to H.
3. H computes $h(r_H||r_U)$ and uses this hash-code to seed a pseudo-random number generator (PRNG) of appropriate characteristics to generate an r-subset $\{i_1, i_2, \ldots, i_r\}$ of $\{1, 2, \ldots, t\}$ — this PRNG could, for example, be based on h. H now sends r_H and X to U.
4. U first checks r_H using the value $h(r_H)$ sent previously. U now recomputes the r-subset $\{i_1, i_2, \ldots, i_r\}$, and continues as in step 2 of the scheme from Section 4.2.

Note that, by sending $h(r_H)$ in the first step, H commits to the random value r_H without revealing it. This prevents either party learning the other party's random value before choosing their own. This, in turn, prevents either party choosing even a small part of the r-subset. Note also that, although this scheme lengthens the protocol, it also slightly reduces the communications complexity, since the r-subset no longer needs to be transferred.

6 Summary and conclusions

A novel unilateral authentication protocol has been presented, which uses symmetric cryptography and only requires public information to be stored at the verifying host. The computational and storage requirements are non-trivial, but may still be potentially attractive to designers of low-cost remote user authentication devices who wish to avoid the complexity of implementing digital signatures.

References

1. R.J. Anderson, F. Bergadano, B. Crispo, J.-H. Lee, C. Manifavas, and R.M. Needham. A new family of authentication protocols. *ACM Operating Systems Review*, **32**(4):9–20, 1998.

2. N. Haller. *The S/KEY one-time password system.* Bellcore, February 1995. Internet RFC 1760.
3. International Organization for Standardization, Genève, Switzerland. *ISO/IEC 10118–3, Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions*, 1998.
4. International Organization for Standardization, Genève, Switzerland. *ISO/IEC 9797–1, Information technology — Security techniques — Message Authentication Codes (MACs) — Part 1: Mechanisms using a block cipher*, 1999.
5. International Organization for Standardization, Genève, Switzerland. *ISO/IEC 9797–2, Information technology — Security techniques — Message Authentication Codes (MACs) — Part 2: Mechanisms using a hash-function*, 2000.
6. L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, **24**:770–772, 1981.
7. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, 1997.
8. C.J. Mitchell and L. Chen. Comments on the S/KEY user authentication scheme. *ACM Operating Systems Review*, **30**(4):12–16, October 1996.
9. M.O. Rabin. Digitalized signatures. In R. DeMillo, D. Dobkin, A. Jones, and R. Lipton, editors, *Foundations of Secure Computation*, pages 155–168. Academic Press, 1978.